12-1989

Attribute Relationships: An Impediment in Automating Schema Integration

Amit P. Sheth
Wright State University - Main Campus, amit@sc.edu

Sunit K. Gala

#5

Attribute Relationships: An Impediment in Automating Schema Integration*

Amit P. Sheth
Bellcore
444 Hoes Lane, Rm 1J-210
Piscataway, NJ 08854 USA
amit@ctt.bellcore.com

Sunit K. Gala**
Database R & D Center
University of Florida
Gainsville FL 32611 USA
vishu@beach.cis.ufl.edu

Success in schema integration depends on understanding the semantics of the schema components (e.g., entity sets, relationship sets, attributes), and the ability to capture and reason about these semantics. An important objective of our work on schema integration is to automate the reasoning as much as possible, and when not possible, depend on the human input and guidance. A key result of comparing the semantics associated with schema objects is that of determining attribute relationships[1]. Once attribute relationships are determined, the task of object class (e.g., entity sets and relationship sets) integration becomes simpler and can be automated to a great extent. In our work, we use a classifier borrowed from the research in knowledge representation (particularly the KL-ONE family of systems) for automatically integrating object classes.

Previous work on attribute equivalence suggests using attribute definitions (or characterizations) to identify attribute relationships. Unfortunately, such definitions are incomplete and can lead to incorrect inferences. We suggest reasoning about the attribute relationships in a semantic space. However, we do not know how the task of identifying attribute relationships can be automated. We also briefly discuss our overall approach to schema integration, but do not discuss object class integration in detail.

1. Background

The problem of schema integration is very difficult if a significant part of the integration process is to be performed automatically. In fact, a completely automatic process is pragmatically impossible because it would require that all semantics of the components in a schema be completely specified. This in turn is not possible because, among other reasons, (1) the current semantic (or other) data models are unable to completely capture a real world state, (2) it will be necessary to capture much more information than is typically captured in a schema, and (3) there can be multiple views and interpretation of a real world state, and the interpretations change with time.

The survey paper [Batini et al 86] discusses and compares twelve methodologies for schema integration work. Subsequently, at least three tools have also been developed [Souza 86, Sheth et al 88, Hayes and Ram 90]. [Batini et al 86] divide the schema integration activities into five steps, namely, preintegration, comparison, conformation, merging, and restructuring. According to our interpretation (which differs from that in [Batini et al 86]), the preintegration activities may involve translation of schemas into a common data model so that they can be compared, and specifying of the global constraints and naming conventions (i.e., activities that may be useful in the comparison step, e.g., specifying a thesaurus that may used for identifying

*PRESENTED AT WORKSHOP ON HETEROGENEOUS DATABASE SYSTEMS, CHICAGO. DECEMBER 11-13, '89.

**This work was supported by and performed at Bellcore.

[1]Attribute relationships includes both attribute equivalences and attribute inclusions. Most of the previous work (e.g., Mannino and Effelsberg 84, Elmasri et al 86, Sheth et al 88) consider only equivalences.

naming conflicts, homonyms, synonyms, etc.).

The comparison activity (also called schema analysis) involves analyzing and comparing various aspects of the component schemas, including identification of naming conflicts (e.g., homonym and synonym detection), domain (i.e., value type) conflicts, structural differences, and constraints differences. Conforming activities are closely tied to the comparing activities since the comparing activities lead to identification of what need to be conformed and it is difficult to compare unless the same information or concept is represented in a similar way in different schemas. One significant result of schema analysis is attribute relationship specification which can lead to specification of how objects classes in the component schemas are related.

We recognize attribute relationship specification and object class relationship specification to be the most critical tasks in the integrating schemas. Determining attribute relationship is an attempt to identify different attributes that represent the similar concepts. Attribute equivalence has been used in [Elmasri et al 86, Sheth et al 88] to identify object classes that may possibly be related by using the heuristic that if some of the attributes of two objects are equivalent, then the object classes may be related[2]. All previous efforts that use attribute equivalence, to our knowledge, identify equivalent attributes manually, or propose heuristics. However, utility of heuristics, called resemblance functions, proposed in [Souza 86] have not been justified using convincing arguments about pragmatics or experimentation. Following attribute equivalence specification, the integrator (a domain expert or the DBA) specifies all relationships between every pair of related classes. The assertion specification is used as an input to a lattice generation or merging activity to generate a single integrated schema (which can be easily automated [Sheth et al 88]). Some of the efforts, particularly those that do not use object assertions for automatic merging, provide a set of operators for the integrator to correlate or integrate the objects in the component schemas to generate an integrated schema [Motro and Bunemann 81].

2. Our Approach

Our approach consists of four phases: preintegration, schema analysis, object integration, and schema restructuring. While these phases can be found in many previous approachs, the details of what is done within each phase, except the preintegration phase, differ significantly. Schema information in our system exists in two models: an extended E-R model (basic E-R with generalization, aggregation, and multivalued attributes) as a user (integrator) model and the CANDIDE model [Beck et al 89] as the system model[3]. The main result of the schema analysis phase is the generation of an attribute hierarchy. In [Mannino and Effelsberg 84, Elmasri et al 86, Sheth et al 88], schema analysis resulted in attribute equivalence specification. In our approach, instead, an attribute hierarchy is produced which, in addition to equivalence, also represents disjointness and inclusion relationships among attributes[4].

The conformation phase takes care of the problem when the same concept is represented using different model constructs. This problem is alleviated in our case because the CANDIDE model supports the orthogonality of constructs. In particular, both entity sets and relationship sets map into object classes, and an attribute domain can be an object class (i.e., it can represent an entity valued attribute).

---

[2]In [Elmasri et al 86, Sheth et al 88], an entity set (relationship set) can be related to another entity set (relationship set) by one of five assertions: *equals, contains/contained-in, overlaps, disjoint-but-integrate, disjoint-and-nonintegrable.*

[3] The CANDIDE model is of utmost relevance to object integration, but is outside the scope of the this paper. We also do not describe the transformation between the Extended E-R and the CANDIDE models.

[4][Larson et al 89] considers the equal, subset, and overlap relationships among domains of all pairs of the attributes, but does not explicitly represent an attribute hierarchy.

In our approach, the object integration is completely automated by applying classification as available in CANDIDE to merge classes from the component schemas to generate a single class taxonomy (discussed in detail in [Sheth and Gala 89]). Thus, assertion specification, except for the cases of *overlap* and *disjoint-but-integratable* is replaced by an automatic reasoner. It is important to note that while the use of attribute equivalence in [Elmasri et al 86, Sheth et al 88] is only as a heuristic to facilitate the object class assertion specification by a human, in our approach, the information contained in the attribute hierarchy is uniformly and automatically exploited for merging various schemas. The inference rules for such exploitation are sound [Beck et al 89].

In some of the previous efforts, the merging and restructuring have not been treated separately (see section 2.8 in [Batini et al 86] for a discussion). However, use of an automatic reasoner (i.e., the classifier) necessarily makes these tasks distinct. The case of *disjoint-but-integratable* can only be discovered and specified by a human, hence the integrator specifies it using a well defined operator in the schema restructuring phase. Furthermore, since we believe that a complete semantics cannot be represented and automatically reasoned upon, the schema restructuring phase provides a set of well defined operators to allow the integrator to make any changes to the automatically integrated schema. The schema restructuring phase involves ad hoc (but well-defined) modification of schemas, although most published work ignores it[5].

3. On Attribute Relationship and Attribute Hierarchy

In the first subsection, we discuss why attribute definitions represent informal semantics. In the second subsection, we comment on the previous efforts on attribute equivalence. In the third subsection, we briefly describe our approach.

3.1. On Informal Semantics of Attribute Definitions

We define the world being modeled in a schema to consist of object classes (in short, classes). Each component of the data model (i.e., either an attribute or a class in CANDIDE, our underlying data model) has an associated *real world semantics* (*RWS*) that represents the intended semantics as perceived by the user. This *RWS* may or may not be captured completely or faithfully by a particular instance of the model as represented in the schema. *RWS* of an attribute is denoted as *RWAS* and that of a class is denoted a *RWCS*.

We say that the class definition represents *formal semantics* for the following two reasons: (a) it is possible, even desirable, to give a formal denotation to object definitions (further details are in [Sheth and Gala 89]) and (b) there is an idempotency between an object (whether individual object or the class) in the model and its real world counterpart belonging to *RWCS*.

In fact, the latter is the *raison d'être* for semantic data models and the knowledge representation schemes, and is implied by the *Knowledge Representation Hypothesis* [Smith 82]. The formal framework allows us to reason about class definitions in a meaningful and automatic way; in particular, it allows us to use the *classifier* defined in the CANDIDE semantic data model.

Reasoning about class definitions invariably gets decomposed into reasoning about the attributes and the constraints on the attributes. Thus computing class relationships is reduced to computing attribute relationships. Now, by applying this reductionist philosophy, we start looking for descriptors of attributes which attempt to define the semantics of the attributes, so that we can automatically reason about attribute relationships. However, attribute definitions represents *informal semantics*, i.e., we don't know how to reason about them, making it difficult, perhaps impossible, to automatically discover attribute relationships. In other

[5]This may be because when the object integration is performed manually, it may be natural to treat the last two phases as a single phase.

words, the function between the real world semantics and the attribute definitions, $f_{att} : RWAS(a) \rightarrow a$, is not injective, i.e., an attribute definition does not uniquely identify its *RWAS* and hence reasoning in the space of all attribute definitions do not imply corresponding reasoning in the space of real world semantics. The problem is that we do not know of an attribute definition that completely specifies its real world semantics.

3.2. Comments on Attribute Equivalence in the Literature

Use of attribute equivalence for schema integration has been discussed in several places including [Mannino and Effelsberg 84, Larson et al 89]. Let us discuss the [Larson et al 89] in further detail. Besides the problem of defining an attribute in reference to the objects (more precisely, in the object class definition), they assume a one-to-one mapping between the attribute definition and the attribute's *RWAS*. They define an attribute in terms of the following descriptors:

Uniqueness
Lower and Upper Bound
Domain (type) and Scale
Static and Dynamic Semantic Integrity Constraints
Security Constraints
Allowable Operations

They furthermore tie the attribute to an object class (entity type or relationship type) definition. The attribute equivalence theory proposed by them essentially says that if there exists mapping functions (that satisfy certain properties) between the domains (i.e., the value set) of two attributes, then the two attributes can be said to be equivalent[6]. We now observe that this attribute definition is *incomplete*. First, the particular set of descriptors used to define an attribute are arbitrary. That is, we can always add some more descriptors or choose a different set of descriptors. These descriptors are mostly chosen based on the information that is traditionally kept in a schema (in part to help in generating mappings) and the implementation aspect of the attribute. Second, the descriptors fail to capture the semantics of an attribute. For example, consider two attributes *person_name* (or attribute *name* attached to an entity type *person*) and *department_name* (or attribute *name* attached to an entity type *department*). We may be able to define a mapping between the domains (types) of these two attributes, and thereby, according to their theory, declare them to be equivalent (note that the issue of whether a mapping is meaningful or not in *RWS* sense is not considered). Unfortunately, this is contrary to the intended semantics of the two attributes. We do not dispute the utility of defining the domain mappings (they have to be defined in order to allow database access from the integrated schemas), but the existence of the mapping does not imply attribute equivalence. Furthermore, we do not know of a particular set of descriptors that capture sufficient attribute semantics to help in identifying attribute relationships. At best, given a set of descriptors, one can define heuristics that may help in discovering attribute relationships. However, such heuristics cannot be fulled depended upon to always make correct equivalences or to detect all meaningful equivalences.

3.3. Attribute Hierarchy Generation

The approach in [Larson et al 89] establishes relationships between all pairs of attributes. We differ with this approach in one significant way. We propose reasoning about the attribute relationships in the semantic space, i.e., based on the *RWAS*, and not in the definition (or model) space. This is in part due to our inability to reason with the informal semantics provided by attribute definition, we propose reasoning about the attributes based on their *RWAS* as follows.

[6]They further define different types of equivalences based on the whether the mapping is bidirectional and whether the mappings hold at the current time or always (at all times). However, these details are inconsequential to our discussion.

For any pair of attributes attributes, a and b, one of the following relationships hold:

(1) a and b are equivalent, or $a \equiv b$,

(2) a contains (is contained-in) b, or $a \supset (\subset) b$,

(3) If neither of the above, then a and b are unrelated.

Our attribute hierarchy is a strict partial ordering of inclusion relationships. Given the above relationships among every pair of the attributes, an attribute hierarchy can easily be generated.

We limit reasoning in our universe to consist of

(1) Assertions about attribute relationships.

(2) Assertions about object definitions, and their relationships.

We can now define a semantic space of RWS[7] for the universe of discourse U to be a union of two (sub-spaces), RWCS and RWAS, as follows:

$$RWS(U) = \{RWCS(c_i)\} \cup \{RWAS(a_j)\}, \qquad 1 \le i \le m, 1 \le j \le n$$

where m is the total number of classes and n is the total number of attributes in U. Let each distinct $RWAS(a_j)$ represent a *semantic point* in *RWAS* space. We further define a *semantic cluster* to be a disjoint partition in the *RWAS* space. The class definitions are denoted by single-place predicates $P(x)$ and attributes are denoted by two-place predicates $Q(x,y)$[8].

Thus, the problem of attribute relationship can be broken into two subproblems:

(1) moving from the space of attribute labels and their characterization to its corresponding semantic space *RWAS*,

(2) generating an attribute hierarchy in the semantic space.

By looking at Figure 1, we can see that the links between the attribute definitions space $S\{Q(x,y)\}$ and the semantic space $RWAS\{Q(x,y)\}$ denote the above mapping. Let

$$f: \{Q(x,y)\} \rightarrow RWAS\{Q(x,y)\}, \text{ and}$$
$$f^{-1}: RWAS\{Q(x,y)\} \rightarrow \{Q(x,y)\}$$

It can be easily seen that the mapping f is *1-1* whereas its inverse is *1-n*. Consider a point $Q_i(x,y)$ in the semantic space. Corresponding to this point we can have many points, $Q_{i_j}(x,y)$, in the attribute definition space.

We now make the following observations:

- Since *emp_name*, *worker_name* and *name* map onto the same semantic point Q_1, they are equivalent.
- It is also the case in our universe that *person_name* and *student_name* which map onto distinct points Q_2 and Q_3 respectively, but are related to Q_1 and are therefore located very "nearby" in the semantic space.
- *emp_num* and *ss_num* map onto Q_4 and Q_5 respectively, and whose distance from Q_1 is "greater" than that between Q_2 and Q_3.
- On the other hand, *dept_name* maps onto the semantic point Q_6 which is *disjoint* from Q_1.

The above suggests a certain classification pattern based upon equivalence and inclusion relationships between attributes as well as their relative meaning. Thus, we can define the following semantic clusters in

[7]The semantic space can be seen as an abstraction of all semantics in the U in the integrator's mind.

[8]One can always translate object definitions in most structured representation schemes (such as semantic data models, semantic networks and frame description languages) into equivalent logic statements which consist of only single- and two-place predicates.

our semantic space:

- $SC_1 = \{Q_1, Q_2, Q_3\}$. This says that the semantic cluster labled as *people_names* for convenience, includes *emp_name*, *person_name* and *student_name* (the set is non-exhaustive and mutually exclusive). However, it is also the case that $Q_1 \subset Q_2$ and $Q_3 \subset Q_2$. This is equivalent to saying that *RWAS(emp_name)* $\subset$ *RWAS(person_name)* and *RWAS(student_name)* $\subset$ *RWAS(person_name)*. Thus, there may be strict partial ordering within a given semantic cluster. But any two semantic clusters are necessarily disjoint. This hierarchy is shown in Figure 2.
- $SC_2 = \{Q_4, Q_5\}$. This says that the cluster SC_2 labled *people_numbers* contains, among other semantic points, *emp_num* and *ss_num*.
- $SC_3 = \{Q_6\}$
- $SC_{12} = SC_1 \cup SC_2$. This says that there is a semantic cluster labeled say *people_identifiers* includes, among other things, the clusters *people_names* and *people_numbers*. Again, this cluster is a non-exhaustive and mutually exclusive union of two semantic subspaces.

We now propose the following methodology to deal with attribute equivalence and inclusion:

(1) Make a list of all attribute names appearing in all the schemas to be integrated.

(2) The integrator (a domain expert or DBA) must now identify various appropriate semantic clusters (this can be done in advance or dynamically when considering a new attribute not belonging to a previously defined semantic cluster)[9].

(3) All sets of equivalent attributes should be determined and each set must be denoted by a unique *RWAS* (i.e., semantic point).

(4) All remaining attributes are then associated with their respectively unique *RWAS*s.

(5) Each *RWAS* is placed in exactly one semantic cluster based on the intuition of the integrator.

(6) Now that each semantic cluster contains a set of unique *RWASs*, the user can specify piece-meal information about inclusion between two *RWAS*s within the same semantic cluster (recall that any two semantic clusters are necessarily disjoint), which can then be assimilated into a strict partially ordered hierarchy. This process of building hierarchy is better than the $O(n^2)$ process of determining relationship among each pair of attributes.

4. Conclusions and Future Work

In this paper we describe our views of attribute relationships, which is the primary result of the most important step of schema analysis in the schema integration process. Our approach involves generating attribute hierarchy. A semantic cluster based methodology helps in generating this hierarchy, but does depend on the human guidance to a significant extent. We show that the earlier work on attribute equivalence do not help in automating this step. The classifier as defined in [Beck et al 89], takes the attribute hierarchy and the object class definitions as the input, and generates a class taxonomy, which is an integrated schema. This integrated schema can be restructured to allow *disjoint-but-integrate* assertions which the automated reasoner cannot decide as well as to allow the integrator to make integration decisions based on the semantics not captured in the attribute and class definitions.

[9]Note that different integrators may make different decisions based on their interpretations, which could lead to different attribute hierarchies. Different attribute hierarchies would result in generation of different class taxonomies (viz., integrated schemas). This is perfectly acceptable, and reflects the fact that there may be multiple integrated schemas for the same set of components schemas.

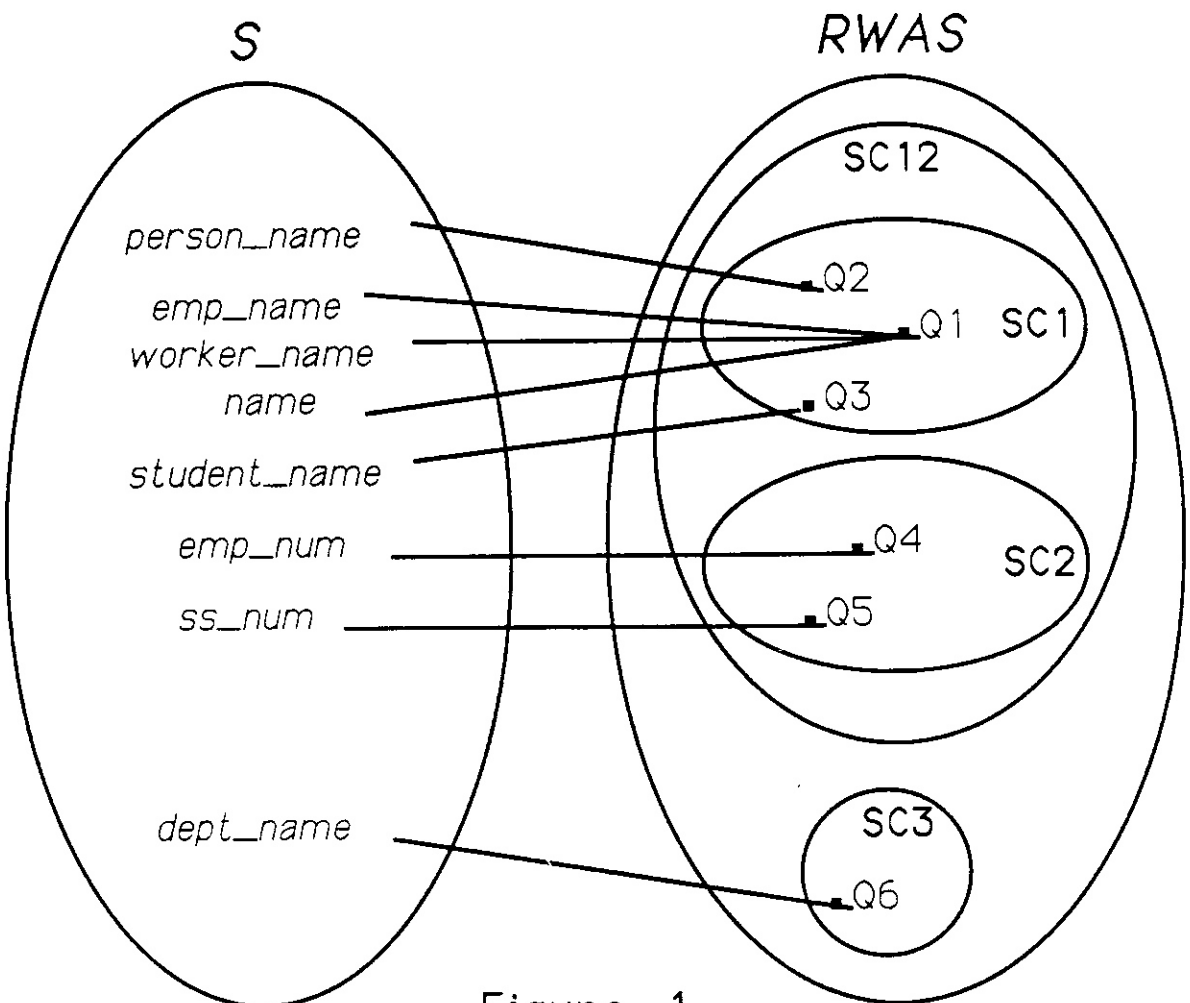

Figure 1

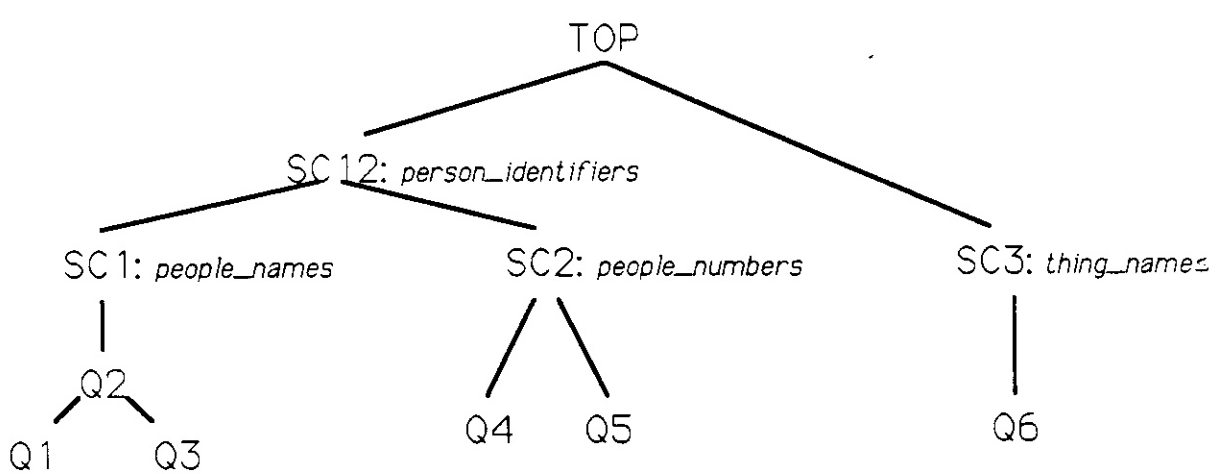

Figure: 2

We are currently developing a software tool based on this approach. The utility of our approach can then be checked using real life examples.

Acknowledgements: We thank Jim Larson for his insightful comments on an earlier version of this paper.

5. References

[Batini et al 86] C. Batini, M. Lenzerini, and S.B. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration," *Computing Surveys*, 18(4), December 1986.

[Beck et al 89] H. Beck, S. Gala, and S. Navathe, "Classification as a Query Processing Technique in CAN-DIDE Semantic Data Model," *Proceedings of the Fifth International Conference on Data Engineering*, Los Angeles, CA, February 1989.

[Elmasri et al 86] Elmasri R., Larson J., and Navathe S.,"Schema Integration Algorithms for Federated Databases and Logical Database Design," Tech. Rep. CSC-86-9:8212, Honeywell Systems Development Division, Minneapolis, MN, April 1986.

[Hayes and Ram 90] S. Hayes and S. Ram, "Multi-User View Integration System (MUVIS): An Expert System for View Integration," *Proceedings of the Sixth International Conference on Data Engineering*, Los Angeles, CA, February 1990.

[Larson et al 89] J. Larson, S. Navathe, and R. Elmasri, "A Theory of Attribute Equivalence in Databases with Applications to Schema Integration," *IEEE Transactions on Software Engineering*, 15(4), April 1989.

[Mannino and Effelsberg 84] M. Mannino and W. Effelsberg, "Matching Techniques in Global Schema Design," *Proceedings of the First International Conference on Data Engineering*, Los Angeles, CA, April 1984.

[Motro and Bunemann 81] A. Motro and P. Buneman, "Constructing Superviews," *Proceeding of ACM SIGMOD*, May 1981.

[Sheth et al 88] A. Sheth, J. Larson, A. Cornellio and S. Navathe, "A Tool for Integrating Conceptual Schemas and User Views," *Proceedings of the Fourth International Conference on Data Engineering*, Los Angeles, CA, February 1987.

[Sheth and Gala 89] A. Sheth and S. Gala, "Resolving Issues in Informal and Formal Semantics for Schema Integration," *in preparation*.

[Smith 82] B. Smith, *Reflection and Semantics in a Procedural Language*, Ph.D. Thesis and Tech. Rep. MIT/LCS/TR-272, MIT, Cambridge, MA, 1982.

[Souza 86] J. de Souza, "SIS - A Schema Integration System," *Proc. BNCOD5 Conference*, 1986.